Science & Technology Facilities Council
Rutherford Appleton Laboratory

Applying the Earth System Grid Security in Heterogeneous Environment of Data Acce Servic

Philip Kershaw

STFC Rutherford Appleton Laboratory

British Atmospheric Data Centre
NATIONAL CENTRE FOR ATMOSPHERIC SCIENCE
NATURAL ENVIRONMENT RESEARCH COUNCIL

NEODC

Centre for Envir ntal Data Archival
SCIENCE AND TECHNOLOGY S COUNCIL
NATURAL ENVIRONMENT R OUNCIL

Coupled Model Intercomparison Project Phase 5

- CMIP5 is a framework for co-ordinated climate change experiments
- Will input into the IPCC 5th Assessment Report (AR5) scheduled for 2013

20 modelling centres	mid-2009: • Simulations Starting
50 numerical experiments	2009: • Model and Simulation Documentation needed
86 simulations (total ensemble members) within experiments	end of 2010: • Data available
6500 years of simulation	early to mid 2012: • Scientific Analysis, Paper Submission and Review
Data to be available from "core-nodes" and "modelling-nodes" in a global federation.	
Users need to find & download datasets, and discriminate between models, and between simulation characteristics.	early 2013!: • Reports

- Software infrastructure under development:

Earth System Grid

InfraStructure for the European Network for Earth System Modelling
is-enes

metafor

Earth System Curator

ESG Access Control Requirements

1. Organisations responsible for model data need the ability to:
   - register users and audit access,
   - keep the user community informed
   - protect finite computing resources
2. But, minimise the technical and administrative barriers to participation
3. Access control must:
   - Layer over heterogeneous mix of individual organizations' existing tools and services
   - whilst at the same time maintaining usability and ease of access.

- *2. and 3. are points of failure for grids / federated systems*

Tackling Heterogeneity

- The problem:
  - Different services
  - Technology stacks
  - Organisational structures
  - Limitations on resources, bandwidth, storage, processing power

Degree of separation of concerns *proportional to* potential interoperability and reusability

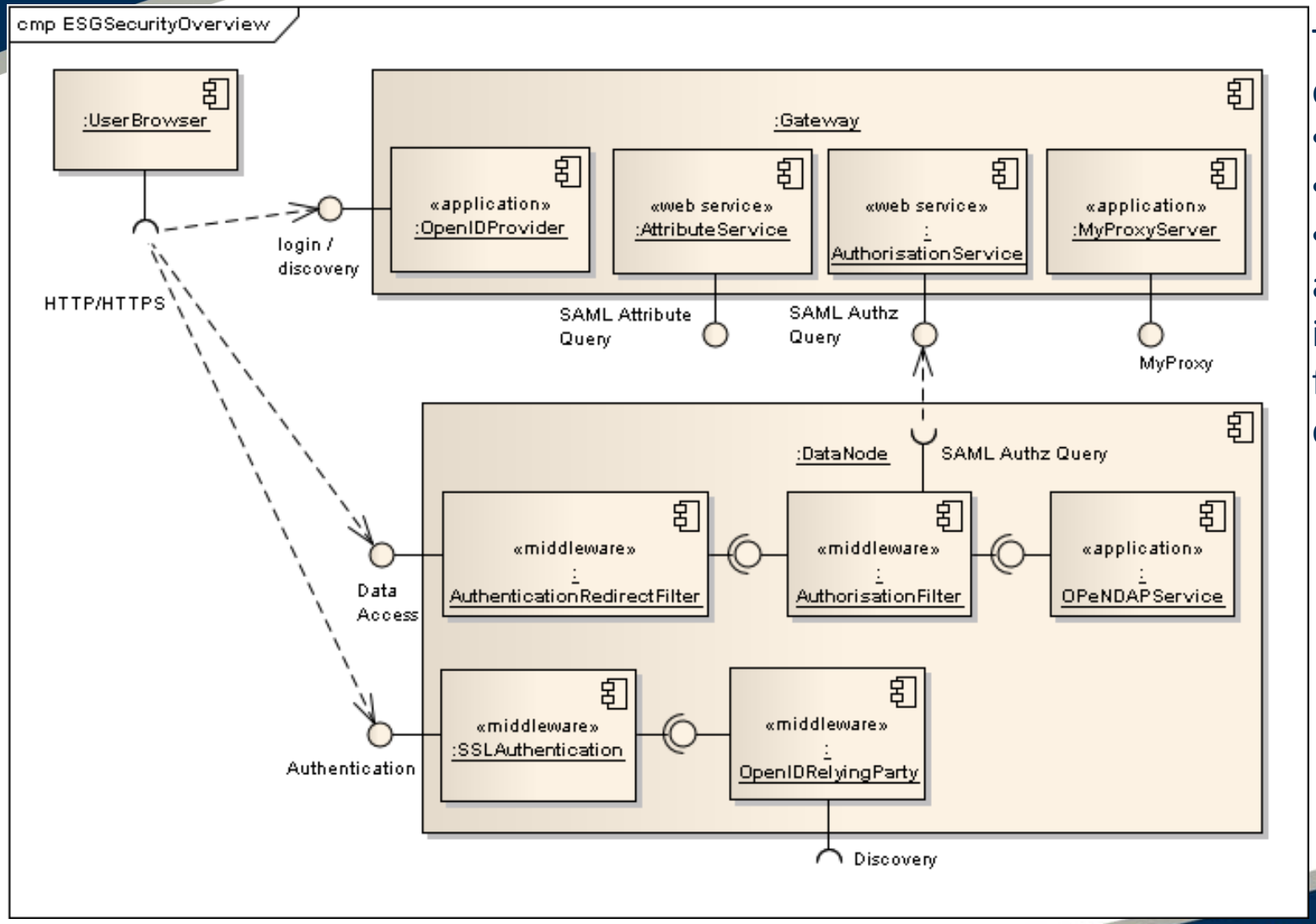
- Some solutions – *separation* through:
  - Web services – SoA
  - but also application middleware
  - REST based principles for Access Control

Separation Requires Interfaces

- Use common libraries or common standards... or both?!
- Answer: common standards, independent implementations
- Common standards:
  - Single sign on: OpenID and MyProxy
  - Attribute Retrieval: SAML 2.0 and OpenID with AX (Attribute Exchange)
  - Authorisation : SAML 2.0
- implementations
  - ESG development team: Java implementation
  - Parallel CEDA implementation in Python

- *Testing across implementations ensures more robust adherence to the standards.*

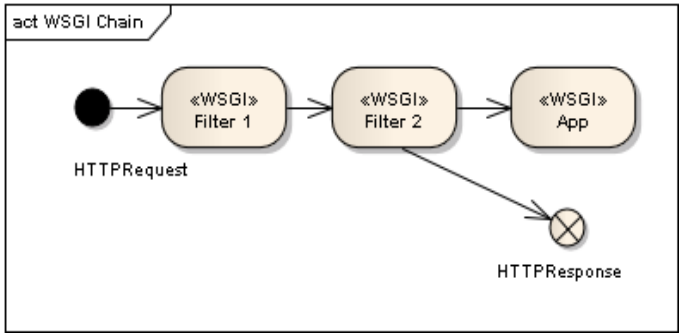
ESG Security Architecture



Two high level components:

- Gateway
- Data Node
- SoA enables mix and match of implementations for each component

Functionality Slicing with WSGI



- SoA – capability to slice up across web service interfaces
- What about the applications themselves?
- Application middleware
  - in Python => WSGI (Web Server Gateway Interface)
  - Akin to Java servlets
  - A web application can be separated into a chain of middleware components each taking a pass over the input request and then passing it on to the next middleware or short circuiting the chain to return a response
  - Slicing based on the functionality being provided

REST and Access Control Policy

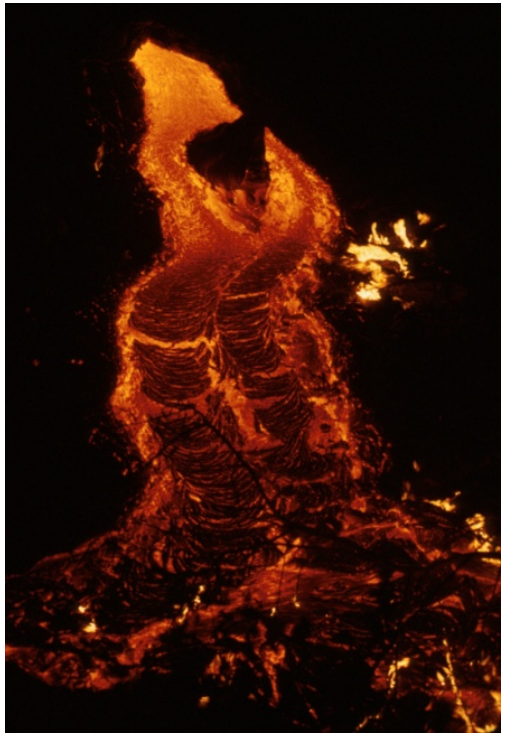
"With URI-based (REST) web services, administrators can apply ACLs [Access Control Lists] to the service itself and to every document that passes through the service, because each of them would have a URI."

"It is much harder to secure an RPC-based system where the addressing model is proprietary and expressed in arbitrary parameters, rather than being group together in a single URI."

- http://www.xml.com/lpt/a/923 REST and the Real World, Paul Prescod, 20 Feb '02

- Different applications and toolkits each with their own security API
- For HTTP, access control policy is determined by the characteristics of a request: the URI, the method GET, PUT etc.
  - Properties which are independent of the specifics of any given API toolkit.
  - This makes it independent of the application's inner workings => separation from the application

Preserving Modularity



An excuse for mentioning volcanoes

- Challenges to modularity:
  - Requirements solidify, implementation beds down and can become brittle – *lava flow antipattern*
  - Developers can prefer application specific security APIs
- Preserve with:
  - Vigorous unit testing
  - Perhaps more importantly integration testing
    - Do the components still fit together OK?!
- Is it worth preserving?

COWS – CEDA OGC Services and ESG Security

- COWS – CEDA OGC Web Services
- Python implementation with the Pylons framework
- COWS WMS secured with generic ESG security filters
  - Can accept SSL and OpenID based authentication
- COWS Client
  - Open Layers based
  - Understands ESG Security enabled COWS WMS:
    - Invoke OpenID based sign in on HTTP *401 Unauthorized* response.
    - *403 Forbidden* => user doesn't have the required access rights
- Demo

Future Plans

- Tackle delegation
  - *MashMyData* Project: user delegates to portal data mash up application which can retrieved secured datasets from other services
  - Proxy certificates
  - *OAuth*
- OGC Authentication Interoperability Experiment (with OpenID)
- OGC WPS (Web Processing Service) implementation and access control policy
  - URI based where possible but
  - policies based on POST'ed request content may be needed
- XACML (eXtensible Access Control Markup Language)
  - Richer functionality for policy expression
  - Standardised policy
  - Python XACML 2.0 implementation recently completed